8-25-1995

Factors Affecting Application Development Productivity: An Empirical Investigation

Roy Schmidt
The University of Science and Technology, Hong Kong

Sunro Lee
Yonsei University, Korea

Factors Affecting Application Development Productivity: An Empirical Investigation

Roy Schmidt
Department of Information and Systems Management
The University of Science and Technology, Hong Kong

Sunro Lee
Department of MIS, College of Commerce and Law
Yonsei University, Korea

Introduction

A recent survey [4] shows that 35% to 50% of software projects had an overrun in terms of the development effort and schedule. Such late deliveries of software tend to cause project backlogs on the order of 374% of current capacity [2]. Many factors have been suggested as root causes for going over the budget, such as inaccurate estimates, inept scheduling, and failure to recognize risks and plan accordingly. Considerable effort has been devoted to the study of software project management techniques to cure these problems [1]. On the other hand, some suggest that we should seek ways to make programmers more productive, just as Henry Ford made auto assembly workers more productive. For instance, Boehm [3] believes that there are opportunities to increase software productivity, and predicts that by 1995 a 20% improvement in software productivity will be worth US$45 billion in the US and US$90 billion world-wide.

As a result, new methodologies and tools for software development have been suggested and developed in order to increase productivity and subsequently alleviate project backlogs. Representative examples include structured analysis and design, computer-aided software engineering (CASE), and fourth generation languages (4GLs).

However, the impacts of these new methodologies and tools on productivity have not been clearly realized, and their use is often controversial [8]. Previous research [6,7] suggests that improving programming productivity requires much more than the isolated implementation of new technologies and policies. Unfortunately, the integration of the new technologies with strategies for managing the software development process, such as project and configuration management, characterization of project teams, and process control, has been largely overlooked [5] as a critical contributor to software productivity improvement. Thus, an integrated analysis, which covers both technological and managerial aspects of software development, is needed.

Since there has been almost no research of this problem, we began with an exploratory study [9] that examined current practices in application systems development, focusing on utilization of productivity tools in information systems (I/S) organizations. The aim of the study was to develop an appreciation of the factors that might affect application

development productivity. We found that an integration of technical and managerial interventions accounted for increased productivity.

In the present study, we follow up on the exploratory study. Using hypotheses developed in [9], we are collecting data from a large sample of companies through a mail survey. Before proceeding to a detailed description of the research in progress, we present some concepts basic to understanding our research approach.

Productivity: The Dependent Variable

Previous studies of software development productivity have tended to focus on *programmer productivity*, that is, they have concentrated on the coding phase of the life cycle. Studies of programmer productivity typically measure productivity in terms of the volume of source code produced per unit of effort, such as lines of code (LOC) per man-hour or function points (FP) per man-hour. Since these measures do not directly account for the effort involved in the other phases of software development, we deem them to be micro-level measures of productivity.

Furthermore, as technologies for systems analysis and design advance and programming experience increases, programming (coding) itself is becoming a routine task. Particularly in business applications, the intellectual challenge and creative opportunities are lessened through the reuse of code and the institution of standardized interfaces. Team-based programming used to consume over 50% of project effort. But recently this effort has gone down to less than 30%, with the remainder of project time redirected toward other phases of the life cycle. Thus, the major portion (70%) of project effort has been ignored in the micro-level productivity measures.

A recent survey [4] shows that the annual backlog has not been reduced for a decade, even though programming productivity (measured in LOC per man-hour) has increased significantly. So to account for productivity enhancement efforts across the entire development life cycle, it is necessary to define and measure application development productivity at the macro (integrated) level. To isolate productivity gains within specific contexts, we also must make our productivity measure relative to the context. Specifically, the annual capacity of an application development group within a given organization can be compared with the annual requirement for systems development and maintenance within that organization. The shortfall in capacity is represented by the annual backlog. An improvement in productivity would be indicated when the long-term trend in annual backlog improves.

In this study, we seek to establish the utility of various methodologies and tools for the enhancement of application development productivity. We believe the most effective methodologies and tools affect the entire life cycle, and thus must be measured at the macro level. If we were to use micro-level measurements in this study, we would seriously misjudge the true contribution of these methodologies and tools, because local gains might be offset by losses in other phases of the life cycle.

The backlog trend makes an excellent indicator for macro-level concerns. Let us cite one example. Suppose an I/S manager institutes an improved practice in analysis, design, or coding. The new technique is somewhat more labor-intensive, such that the "LOC count" is worse -- the micro-level measure indicates a decline in programmer productivity. But the new technique produces better-designed software resulting in significantly fewer maintenance/enhancement requests over the first few years of software use. The net effect would be some reduction in backlog if the sum of the labor consumed in coding and maintenance is less than under the previous regime. At the macro level, we observe a gain in productivity.

The Independent Variables

We can classify productivity-enhancement measures into two types, technical and managerial. The technical measures aim at improvement through refined techniques and technology, such as 4GLs and CASE. The managerial category includes approaches such as outsourcing, end-user computing (EUC), and process controls to solve the productivity problem through non-technical means.

We can further classify the productivity-related methodologies and tools according to their impact. Some measures are narrowly focused (micro) solutions, such as automatic code generators. Others have a broader (macro) effect, such as a new life-cycle methodology or integrated CASE.

Thus, we place current practices into one of four categories: technical micro-level (4GLs, code reuse, packaged software), technical macro-level (CASE, open systems, object-orientation), managerial micro-level (outsourcing, contract programmers, end-user computing, dual-track careers), and managerial macro-level (new development methods, process control). In this way, we can examine individual affects and categorical effects in the study.

We also test for other factors that might contribute to the success or failure of the above methods and tools: short and long-term planning, organizational structure of the I/S department, personnel turnover, education level and experience level of programmers and analysts, programming languages used, and allocation of effort among analysis, design, coding, and testing. These factors were uncovered in our exploratory study [9], but only one (long-range planning) seemed to have any effect on the efficacy of productivity enhancement.

Design of the Study

Questionnaires have been sent to a random sample of organizations in Hong Kong having an in-house I/S development capability. Data on the above variables is collected, along with some demographic data to help determine the generalizabilty of the study. Due to space limitations, we cannot include a copy of the questionnaire in this paper, but a copy can be obtained from the authors upon request.

In our first study, we interviewed I/S managers and project managers in 18 companies across a broad range of industries. In each of the 18 companies, the I/S department had over 50 employees. After careful analysis of the interview data, we determined that a combination of the use of CASE tools and long-range planning were the primary factors contributing to better productivity. That is, in those companies using CASE and having active, long-range I/S planning the annual backlog was under control despite a steady increase in demand for new projects. Since we class both long-range planning and CASE as macro-level practices, we believe such practices have greater promise for increased productivity than do micro-level activities.

So based on the results of the first study, we have four principle hypotheses that we will test directly with the data from the current study.

H1: The use of macro-level practices will be positively associated with better productivity.

H2: The effect of the macro-level practices will be affected by the use of long-term planning.

H3: The use of CASE will have the greatest positive effect on productivity.

H4: The effect of CASE on productivity will be affected by the use of long-term planning.

A combination of nonparametric statistical techniques and ANOVA will be used to test these hypotheses.

Contribution of Study

Research concerning application development productivity at the macro level is conspicuously sparse. Through a series of studies, we aim to build a theoretical base for future investigation of the productivity problem and its solution. We are still at the exploratory, theory-building stage of research. By presenting our research-in-progress in this forum, we hope to attract the interest of other researchers to this line of investigation. We believe the development of theory concerning application development productivity has great potential for improving future practice.

I/S managers currently have a micro focus on productivity improvement. Through our study, we hope to build evidence to convince practitioners to move to the use of macro-level methods and tools, and to have greater regard for I/S planning.

References

[1] Abdel-Hamid, T., and S. E. Madnick, *Software Project Dynamics: An Integrated Approach.* Englewood Cliffs, NJ: Prentice-Hall, 1991.

[2] Alloway, R. M. and J. A. Quillard, "User managers' systems needs", *MIS Quarterly* 7(2), 1983, pp. 27-41.

[3] Boehm, B. W., "Improving software productivity", *IEEE Computer* 20(8), 1987, pp. 43-58.

[4] Genuchten, M. V., "Why is software late? An empirical study of reasons for delay in software development", *IEEE Transactions on Software Engineering* 17(6), 1991, pp. 582-590.

[5] Henderson, J. C. and S. Lee, "Managing I/S design teams: A control theories perspective", *Management Science* 38(6), 1992, pp. 757-777.

[6] Jeffrey, D. R., "Software engineering productivity models for management information system development", in R. L. Boland and R. A. Hirscheim (eds.), *Critical Issues in Information Systems Research.* New York: Wiley, 1987, pp. 113-134.

[7] Jones, C., *Programming Productivity*, New York: McGraw-Hill, 1986.

[8] Kemerer, C. F., "How the learning curve affects CASE tool adoption", *IEEE Software* 9(3), 1992, pp. 23-28.

[9] Lee, S., and R. C. Schmidt, "Improving Application Development Productivity in Hong Kong", forthcoming in J. Burn (ed.) *The Fire of the Dragon: The Competitive Role of Information Technology in Hong Kong*, London: McGraw-Hill, 1995.